AR-010-281

DSTO

Underwater Acoustic Imaging: A Computing Hardware Approach to Rapid Processing

David G. Blair

DSTO-TN-0099

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Underwater Acoustic Imaging: A Computing Hardware Approach to Rapid Processing

David G. Blair

Maritime Operations Division
Aeronautical and Maritime Research Laboratory

DSTO-TN-0099

ABSTRACT

High-resolution underwater acoustic imaging using multi-element arrays implies a large computational load. For a three-dimensional viewing volume resolved into 3×10^9 voxels (volume pixels), with 4000 elements, the computations needed are around $9\times(1.2\times10^{13})$ floating-point operations. This report develops one of the more promising options for computing the full image. First, parallel computation is used to deal with the different sensor elements simultaneously, when calculating the address of the appropriate instantaneous voltage at the sensor element—or, equivalently, the calculation of the round-trip distance travelled by the acoustic pulse. This calculation requires, in a typical near-field situation, the computation either of a square root or of a fifth degree polynomial. This polynomial allows increased parallelism. Second, the summation in the beamforming is likewise done with a high degree of parallelism. A machine with the above design, with 10^9 clock cycles per second, would compute the entire image in roughly 6 seconds. Cost and availability are not investigated.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001 Australia

Telephone: (03) 9626 7000
Fax: (03) 9626 7999

AR-010-281
September 1997

APPROVED FOR PUBLIC RELEASE

Underwater Acoustic Imaging: A Computing Hardware Approach to Rapid Processing

Executive Summary

As part of Australian defence policy to counter sea mines, the need has been recognised for an enhanced capability to identify objects that have been classified as minelike. Because the medium is often too turbid for optical methods to work, DSTO is working with Australian industry to develop a high-resolution underwater acoustic (sonar) imaging system, intended to 'see' detail of size 1 mm at a range of 1 m. The system has successfully passed the concept demonstrator stage.

The signal processing or beamforming for the system is being developed. The processing involves, for each of 3×10^9 volume pixels (voxels), a summation of 4000 voltages, one from each sensor element, taken from that element's voltage stream at the appropriate time. Therefore (ignoring possible solutions that reduce the number of operations) the number of operations required is at least $4000\times3\times10^9$, and is in fact about nine times that number. Thus a 1 Gflops computer (10^9 floating point operations per second) would require 30 hours to produce the image, unacceptably long by far. A practical computer requires a further 'factor of safety' of say 2.

This report develops one of the more plausible options for computing the image in a greatly reduced time, while retaining the full three-dimensional image: that is, no loss in volume imaged, in resolution or in signal-to-noise ratio. The first feature of this option is that parallel computation is used to deal with the different sensor elements simultaneously, when calculating the address for the retrieval of the appropriate instantaneous voltage at the element—or, equivalently, when calculating the round-trip distance travelled by the acoustic pulse. It is shown that this calculation, when done with sufficient accuracy, requires, in this near-field situation, the computation either of a square root or of a fifth degree polynomial (unless there is a method that increases efficiency by reducing the degree from 5). The polynomial is preferable, particularly as it allows increased parallelism. The second feature is that the summation in the beamforming is also done with a high degree of parallelism.

A machine with the suggested design would output one new voxel in each clock cycle. Thus a machine having 10^9 clock cycles per second would (including the 'factor of safety' of 2) compute the entire image in 6 seconds, apart from the time required for dechirping and related operations. Methods are described for keeping the latter time relatively small. Cost and availability are not investigated.

Author

David G. Blair
Maritime Operations Division

David G. Blair is a senior research scientist based in Sydney. He obtained his PhD for work on electronic properties of disordered materials from the University of Sydney in 1967. He has since worked, in Canada and Australia, on dislocations in solids, correlations in fluids, and electromagnetic exploration for minerals. After 16 years on the lecturing staff in physics at the University of Technology, Sydney, he joined DSTO in 1990 to work on sonar.

Contents

1. Introduction

As discussed in an earlier report [Jones 1996] and in a companion to the present report [Blair and Jones 1997; see also references therein], an innovation program was commenced in 1992 to develop an underwater acoustic imaging system capable of imaging in turbid waters. The system has a target of a range resolution of 1 mm and an angular resolution corresponding to 1 mm at 1 m range. Following feasibility studies, a three-stream approach was adopted in the acoustic sensor innovation program, leading to a successful demonstration of the technology in 1995. The next component of the program is the development of a signal processing and display strategy.

Options for the signal processing phase have been reviewed by Blair and Jones [1997]. The goal of that report is to determine how to calculate the full three-dimensional (3-D) image in the shortest time or, if that time is too long to be useful, to develop the best strategy for obtaining the an image over a reduced volume embracing the object of interest. The authors suggest that there are three broad options for bringing the process of image formation within technically possible bounds. One of the three broad options suggested is the 'hardware' option: the achievement of increased speed via a faster serial computer, a parallel computer or some physical analogue. The report notes that parallel computation, and in particular the treating of sensor elements in parallel, is an option with some promise.

The present report develops in some detail the option of treating the elements in parallel, while adhering to the aim of calculating the full image (at the full resolution and signal-to-noise ratio). Section 2 gives the basic theory and presents the basic problem of computing time. Section 3 begins the discussion of the present attempted solution, describing the large-scale, and some of the smaller-scale, features of the proposed parallel computer. Continuing that discussion, Section 4 is devoted to the calculation of the address for fetching the relevant voltage. Using an expansion that begins with the far-field expression for the path length, an approximation suited to the operational system is found, involving a fifth degree polynomial. Further details of the hardware are then filled in. For the proposed solution, Section 5 gives the pipeline length and an estimate of the overall computing time. Apart from broad comments on the likely cost, the costs and availability are not explored at this stage. The estimates of computing time assume that the dechirping of the recorded signal and the calculation of the quadrature part take a negligible time and cost. Section 6 discusses how such a negligible time and cost might be achieved. Section 7 discusses possible future work, while Section 8 gives conclusions.

2. The Problem

2.1 The System and its Parameters

The system studied is shown in Figure 1. The electrical waveform into the projector is a linear chirp signal. The receiving array consists of tiles, each containing an identical number of elements (sensors). The overall array approximates a random array. After

sampling at the sensors, the voltage stream at each sensor is dechirped by crosscorrelating it with the effective[1] signal into the projector. Signal processing continues with the calculation of a quadrature voltage stream. The major signal processing task, beamforming, is then carried out by summation over the elements. The theory of this calculation is given in Section 2.2 below.

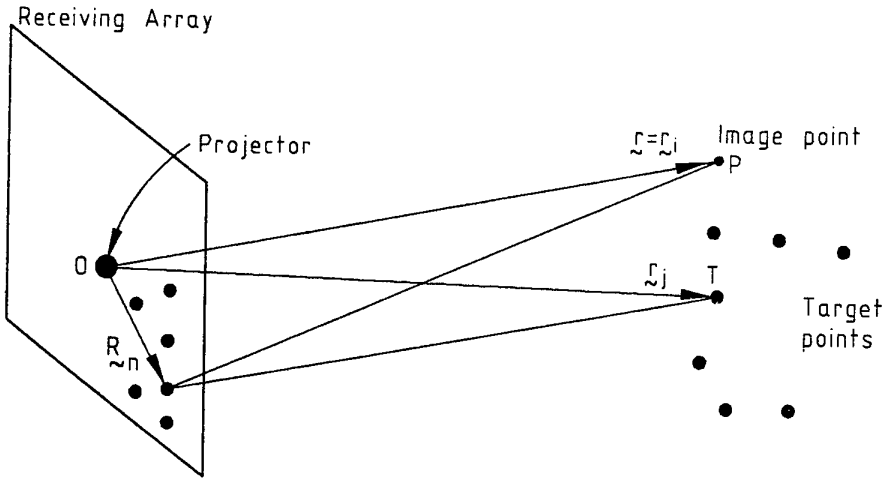

Figure 1: The geometry of the imaging system. The acoustic wave emanating from the projector at O is scattered from a typical target point T and received at the n th array element located at $\mathbf{R}_n$. *The image point P is a typical point at which the image intensity is to be calculated.*

As discussed by Blair and Jones [1997], the following are a typical set of parameters for the beamforming problem:

central frequency	3.5 MHz
bandwidth (approx.)	1 MHz
diameter of array	430 mm
number of elements	4000
number of tiles	100 (40 elements per tile)
sampling rate	20 MHz
number of samples in received chirp signal	80×10^3
number of voxels	$1000 \times 1000 \times 3000 = 3 \times 10^9$ (1000 for each angle, range 1 m to 4 m at 1 mm steps)
speed of sound	1500 m/s
effective signal	single shaped, linear chirp, interval 3–4 MHz

[1] Due to the nonuniform frequency response of the transducers and of the medium, the effective input signal differs from the actual electrical input.

2.2 Beamforming: Theory

The basic equation for the beamforming is:

$$\Pi(\mathbf{r}) = \sum_n w_n \, E_n\left[L(\mathbf{r},n)/c\right] \tag{1}$$

(This method of beamforming in the near and very near field has been briefly described by Knudsen [1989].) In Equation (1), $\Pi(\mathbf{r})$ is the in-phase image amplitude at the 'image point' $\mathbf{r}$, which eventually must be combined with the quadrature part to give the image intensity. (Alternative methods, in particular peak detection, are possible.) Also w_n is the weight (shading) assigned to the nth element, while $E_n(t)$ is the voltage signal at the nth element at time t. $L(\mathbf{r},n)$ is the round-trip distance from the projector (at the origin) to $\mathbf{r}$ (the image point) and back to the nth element (at $\mathbf{R}_n$), given by

$$L(\mathbf{r},n) = r + \left|\mathbf{r} - \mathbf{R}_n\right| \tag{2}$$

(Fig. 1).

In Equation (1), $E_n(\cdot)$ must be taken to mean the voltage after dechirping. It is further assumed that a discrete Hilbert transform of $E_n(t)$ is then performed to yield $E_{qn}(t)$, the quadrature part of the voltage, to be stored along with $E_n(t)$. Furthermore both functions are then multiplied by the weight w_n to yield new stored values

$$F_n(t) = w_n E_n(t) \quad \text{and} \quad F_{qn}(t) = w_n E_{qn}(t), \tag{3}$$

the weighted voltages.

To express Equation (1) in a form that shows most clearly the computing problem, consider the image point to be at the centre ($\mathbf{r}_i$) of the ith voxel. Then (1) may be written as

$$\Pi_i = \sum_{n=1}^{4\times10^3} F_n\left[t = T(i,n)\right] \qquad i = 1,\cdots,3\times10^9. \tag{4}$$

Here $T(i,n) = L(i,n)/c$ is the round-trip time, given by

$$\begin{aligned} T(i,n) &= \left[r_i + \left|\mathbf{r}_i - \mathbf{R}_n\right|\right]/c \\ &= \left[r_i + \left(r_i^2 - 2\mathbf{r}_i \cdot \mathbf{R}_n + R_n^2\right)^{1/2}\right]/c \ ; \end{aligned} \tag{5}$$

while $\Pi_i = \Pi(\mathbf{r}_i)$.

2.3 Dechirping; The Quadrature Part

Before proceeding to the beamforming proper, two main operations and one small operation must be performed on the time-stream of voltages recorded for each element. As a preliminary, we note that in this report the term 'in-phase' signal means the actual or physical signal, $s(t)$ say. The 'quadrature' signal is the Hilbert transform $\hat{s}(t)$ of $s(t)$, which is essentially $s(t)$ but with the phase of each Fourier component

shifted by 90°. Both these signals are real variables. The analytic signal is $s(t) + j\hat{s}(t)$, where $j = \sqrt{-1}$. These concepts are discussed by Rihaczek [1985, pp. 15–20], and again in Section 6.2 of this report.

As the first of the two main operations referred to above, crosscorrelation or dechirping must be carried out, to obtain a signal that, upon beamforming, will yield good range resolution. Second, from the in-phase time-stream of voltages, the quadrature stream must be derived. This is done because, as is known from medical ultrasound practice, the desired image intensity is actually the envelope of a megahertz signal, and one way to calculate the envelope is to make use of the quadrature part. We shall call the obtaining of the quadrature part 'transforming', because one method of obtaining it is to Hilbert transform the in-phase part. The third and small operation is to weight the voltages as in Equation (3).

We now estimate the number of operations required, anticipating the results of calculations carried out in Sections 4.5.2 and 6.3. To dechirp, to obtain the quadrature part and to weight the elements, requires approximately $10 \times 4000 \times (8 \times 10^4) \times \log_2(8 \times 10^4)$ multiply or add operations (MAD operations) (Section 6.3). This is very small compared to the $9 \times 4000 \times (3 \times 10^9)$ operations (see Section 2.4 and, for the '9', Section 4.5.2) required to beamform for a full image. Hence the dechirp/transform stage should not be critical in determining the minimum time for production of a full image.

In view of this fact, the report will make beamforming proper its prime concern. In Section 6 we return to the dechirp/transform stage, describing two ways in which the computing time goal may be met. Further analysis of alternatives for that stage appears to require detailed costings of components.

2.4 The Central Problem: Beamforming Proper

From Equation (4), the summations in the calculation of the image amplitudes require a total of $4000 \times (3 \times 10^9) = 1.2 \times 10^{13}$ operations; double this to include the quadrature part. In addition, there is the calculation of the address for fetching the voltage to be retrieved; we shall call this calculation 'addressing'. It is essentially the calculation of the round-trip distance (Eqn 5). This requires the computing time of several MAD operations for every one add in the summation, because the image point is in the near field. Anticipating results from Section 4.5.2, we conclude that addressing plus quadrature cause the above figure of 1.2×10^{13} to expand altogether to approximately $9 \times (1.2 \times 10^{13})$ operations. Then a 1 Gflops computer (10^9 floating point operations per second) would require 30 hours to produce the image, unacceptably long by far. And further overheads are required by a practical computer: first, due to the lack of ideal matching between the algorithm and the computer, and second, because there are additional items such as display. These overheads require a further 'factor of safety', which we shall take to be 2. The computation time becomes 60 hours.

(The term Gflops is used in this report to refer to operations, whether they be floating point or integer operations. A consequence is that, if integer arithmetic is chosen for the beamforming calculation, as is tentatively recommended later in the

report, that choice makes the speed of '1 Gflops' easier to achieve, at least on the face of it.)

Of the many options for speeding up the calculation [Blair and Jones 1997], this report develops one potential solution, in which many of the calculations are done in parallel. This is a suboption of 'hardware approach', one of three broad options identified in that report.

3. General Architecture of 'Benchmark' Method

3.1 Introduction

In Section 3 we begin the description of the method we describe as the 'benchmark': the performance and cost of methods proposed subsequent to this report can be usefully expressed by comparing them with this method. The calculation of the address for fetching the relevant voltage in beamforming is a large topic, and is therefore postponed to Section 4. The present section treats the remainder of the method.

From Equations (4) and (5), assuming that the voltages in a time-stream are stored sequentially, the calculation of the address is essentially the calculation of the round-trip time. It is assumed that no interpolation is performed, that is, when Equation (4) calls for a particular value of time, the nearest time at which a sample voltage was taken is used.

3.2 Basic Concept

The basic idea is to treat the 4000 elements in parallel.[2] To implement this idea, it is envisaged that there be 4000 computing units, each storing the weighted voltages from one sensor element (both in-phase and quadrature, after dechirping). These units will be referred to as *store-address-fetch (SAF) units*. These 4000 units, together with two summing units, or *summers*, form what will be called the *'parallel computer'*, which performs the core of the beamforming calculation.

3.3 Top-Level Architecture

Figure 2 shows the overall flow of the signal processing. The work of processing is divided between electronics (at the front end), a 'master' computer and the parallel computer.

[2] Treating the *elements* in parallel is preferred to treating a subset of the *voxels* in parallel, because on occasions, several voxels would be reading from the same address, posing a new problem.

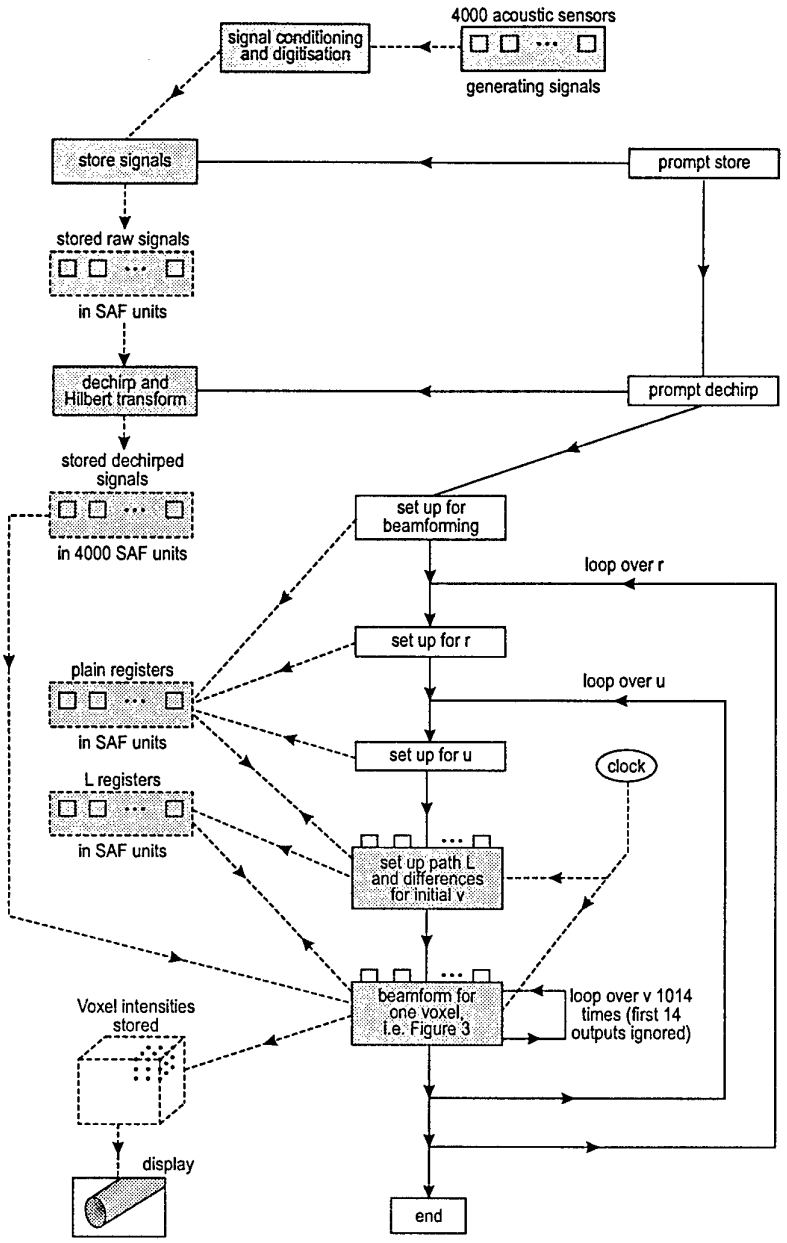

Figure 2: *Overall flow of the computation, incorporating the FFT method of dechirping (Section 6.3.2). Unshaded boxes represent the master (serial) computer. Shaded boxes are used to represent represent both pure electronics (the top two boxes) and also the parallel computer. Solid lines represent the flow of control in the master computer and in the beamforming proper. Broken lines represent other flows (including electrical signals and transfer of data). Solid boxes represent an operation. Broken-line boxes represent stored data or an object upon which an operation can be performed. The figure is discussed in the text (Section 3.3).*

The *master computer* maintains synchronisation, and performs any overseeing necessary during the passing of the raw voltage signal stream from each element into the corresponding SAF unit for storage. It also has a role in initiating each run carried out by the parallel computer, and in retaking control upon the ending of each such run. Finally, the master computer stores the voxel intensity values as they become available, and handles display.

In a serial computer, the beamforming calculation for one voxel would be placed inside three nested loops to cause the system to step through all the voxels. Except for the effect of pipelining in the parallel computer, this nesting arrangement continues to hold in the present situation. From outer to inner loop, these three nested loops are over r (range), u and v respectively. Here u and v are direction cosines to be defined in Section 4.2. For present purposes they may be thought of as angular variables. Just as is the case with spherical polar coordinates (r,θ,ϕ), the triad (r,u,v) specifies a point in 3-D space, in this case the centre of a voxel.

Of the three nested loops (over r, u and v), the master computer controls the outer two loops; in the process it sets up the values of r and u (and possibly a few other parameters) in each SAF unit of the *parallel computer.* (Note that all these parameters are common to the 4000 units.) We use the term 'plain registers' to describe the locations for these parameters in each SAF unit (Fig. 2). Before entering the innermost loop, the parallel computer, acting as 4000 serial computers, initialises certain other parameters for the row of voxels (r and u constant). These parameters are initial values required for calculating the path lengths L throughout the inner loop. These do vary from element to element, and hence from unit to unit. The operation is represented by the uppermost of the two hatched boxes in the central column of Figure 2. These initial parameters are stored in the SAF units, in 'L registers' (Fig. 2). As will be seen in Section 4.5, the values in the L registers are actually updated every time that v is incremented as the inner loop proceeds.

Once the inner loop (loop over v) is set up, the parallel computer cycles autonomously around the that loop (the only outside control being from the master clock, necessary to keep the parallel elements within it synchronised). This operation is represented by the lower hatched box in the central column. The parallel computer is discussed further below (Section 4 and the remainder of Section 3). While we have spoken of a loop over v, the calculations for values of v that are near neighbours actually overlap in time, since the procedure is pipelined.

3.4 Parallel Computer: Overall

The parallel computer performs four operations shown in Figure 2 (the boxes shown hatched within a continuous outline). Of these, the central beamforming operation (lowest of the four boxes) is shown in more detail in Figure 3. Note that Figure 3 does not show all that the parallel computer does; in particular, it is limited to showing what happens in a single clock cycle during the loop over v, a cycle that is not near the start or end of the loop. We now discuss in turn the parts of the parallel computer shown in Figure 3. The other, less central parts or aspects will be discussed in later sections.

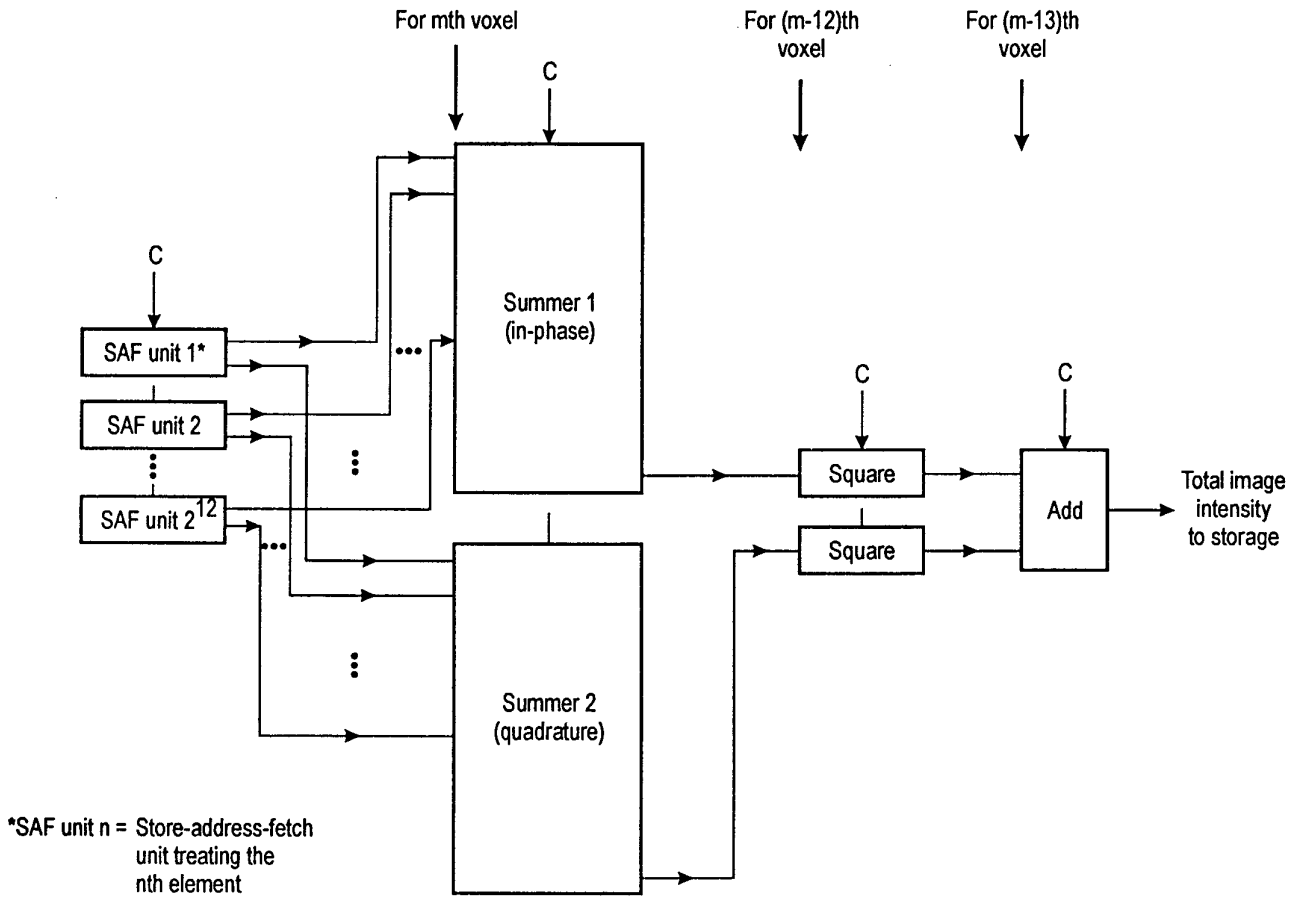

Figure 3: *Overall arrangement for calculating the total image intensity. Initial setting-up and termination for the row of voxels are not shown. During the one clock cycle shown, while the operations in the first round within the summer are processing one voxel (the mth) (using values fetched from the SAF units during the same cycle), the add on the right is processing the $(m-13)$ th voxel. More detail is shown in later figures. C refers to the master clock.*

3.4.1 The Store-Address-Fetch Units

As the first major part of the parallel computer, there are $2^{12} = 4096$ store-address-fetch (SAF) units (Fig. 3), one for each element. Here, for ease of presentation, we suppose that the number of elements has been increased from 4000 to the next power of 2; the additional 'elements' may be supposed to have a null voltage. As we shall see, each SAF unit requires memory for around 2×(80 000) integers. For a ball-park figure, let us assume that each integer requires 8 bits, or 1 byte; then each unit requires 160 kilobytes. The (weighted, dechirped) in-phase voltages, at the various sampling times, for the nth element, are stored in sequential addresses in the nth unit; the quadrature voltages likewise are stored in sequential addresses in that unit. These units also contain the plain registers and the L registers, shown hatched on the left-hand side of Figure 2. Each unit computes the address from which the appropriate in-phase voltage is to be taken for forwarding to the summer; the address is essentially the delay time. The same address plus a constant offset serves to retrieve the quadrature part. The details of the addressing are postponed to Section 4.

At the end of each clock cycle, all the voltages (both in-phase and quadrature) appropriate to some voxel, say the m th, are being output by the SAF units. At the end of the next cycle, the voltages appropriate to the $(m+1)$ th voxel are being output. Cycles near the beginning and end of the loop over v, including initialisation of the loop, require special attention and will be postponed to Section 4.

Once the dechirped voltages have been fed in (at the beginning of the processing of a 'ping'), the only inputs into each SAF unit are the clock and occasionally (at the beginning of each loop over v) an initiating signal from the master computer. Otherwise each SAF unit runs autonomously. The initiating signal is not shown in Figure 3, as this figure shows what happens in a single typical clock cycle. The reasons why no other inputs are need are twofold. First, the figure shows a *typical* cycle, so that the inputs needed in initialising for the loop are not relevant. Second (as will be discussed in Section 4), v is updated internally, so that there is no need to input each new value of v as the loop proceeds.

3.4.2 Other Units

As the second major part of the parallel computer, there are two special summers (Fig. 3): one to sum rapidly all the 2^{12} in-phase voltages and one to sum the quadrature voltages. These summers will be discussed in Section 3.5.

Besides these major parts of the parallel computer, it is necessary for the outputs of the summers to be squared and then added to yield the total image intensity for a given voxel. The square-and-add unit is a combination of three units (Fig. 3), the computing capacity of which is minuscule compared to either the combination of SAF units or the summers. The output is the image intensity of the voxel, to be sent for storage in the master computer (Fig. 2). Each square and each add are assumed to take one clock cycle.

3.4.3 General

The clock pulses are also input into the summers and into the square-and-add unit, to maintain synchronisation throughout. (As will be seen, an initiating signal into these seems unnecessary.)

The procedure is pipelined, as will become clear in subsequent sections. As a result (Fig. 3), the signal being output from one of the summers in any clock cycle refers to an 'earlier' voxel than the signals being output in the same clock cycle from the SAF units; 12 voxels earlier in fact. The latter figure comes about because the summation for a voxel requires 12 cycles, as discussed in Section 3.5. (Twelve is the length of the pipeline within the summer box in Fig. 3.)

3.5 Parallel Computer: The Summers

A pair of identical special summers (Fig. 4), one for the in-phase and one for the quadrature part (Fig. 3), is to sum the weighted voltages. The special summer achieves high speed through a parallelism that reduces the number of clock cycles to calculate the in-phase sum for a given voxel, from $2^{12}-1$ to $\log_2\left(2^{12}\right)=12$. This summer is duplicated for the quadrature part.

Let us follow the additions necessary for processing the m th voxel (the in-phase part, say). In step 1 (top left of Fig. 4), the voltages are added in pairs, 2^{11}=2048 pairs in all. In step 2 (next step down in the figure), the resulting quantities are added in pairs (2^{10} pairs), but one clock cycle later. And so on. Figure 4 is set up to show what happens in one clock cycle, and it therefore shows the 12 of these steps occurring at once, but for *different voxels*. While the weighted voltages for the mth voxel (top left of the figure) are being added in pairs, the sum for the $(m-12)$th voxel (bottom left) is being fetched for the squarer. Each of the two summers is performing 4095 additions simultaneously. With this *double parallelism* (parallelism over 2^k operations within each step, and parallelism over the 12 steps), the overall effective time for the summation becomes *one clock cycle* per total intensity output.

The length of the pipeline within the summer—i.e. the number of clock cycles from entry into the summer to exit—is 12. This is seen from the sequence of numbers, 2^1 to 2^{12}, on the left side of Figure 4. To avoid partial operations in the count, we adopt here the convention that the initial fetch of the 2^{12} numbers is counted as occurring in the summer rather than the addressing unit (Section 4.5.2). Similarly the final fetch is counted as occurring in the square-and-add unit. The pipeline length of 12 is reflected in Figure 3, where the signals into and out of the summer refer to voxel numbers differing by 12.

3.5.1 Alternative Arrangements

A second and a third arrangement for summing the 2^{12} numbers are worth discussing—alternatives to the above 'recommended' method. In all three

arrangements, an incoming set of 2^{12} numbers follows the same course through time as follows. In the first clock cycle, 2^{11} additions (the first 'round' of additions) are performed, in the second, 2^{10} additions, and so on, somewhat as in a tennis tournament. Twelve of these rounds, taking a total of twelve cycles, are required to complete the summation.

In the second arrangement, considering the in-phase part only, there is a summer which can perform the 2^k operations within any one round in parallel, but which can execute only one round at once. For this arrangement the effective time for a summation is 12 cycles, not one. In that case the two summers (in-phase and quadrature), rather than the addressing hardware, become the limiting factor on the computing time (assuming that the addressing is done in an effective time per voxel of less than 12 clock cycles).

In the third arrangement, a summer of the same design as in the second arrangement is used, but computing speed is regained by the inclusion of 12 times as many summers; then there are 12 summers for the in-phase part and 24 altogether. Consider the in-phase summers: each summer is dedicated to one voxel during the course of 12 cycles. One cycle after one summer begins on one voxel, the next summer begins on the next voxel. Each of the 12 summers operates one cycle out of step with the next. During each cycle, one of the 12 summers finishes with its current voxel and moves on to a new one. The effective time is again reduced to one clock cycle per voxel output.

Let us compare the amount of hardware required for the three methods, by comparing the number of adders needed, putting aside the extra hardware that may be required to control the logic. (In this report an 'adder' is a device that simply adds two numbers.) Compared to the first arrangement, which requires (for the in-phase part) 4095 adders, the second arrangement requires only half of this, 2048 adders, since the adders used in the first round can be reused in subsequent rounds. The third arrangement requires 12×2048 adders. Thus the hardware requirements of the first, second and third arrangements are in the ratio 2:1:12. Hence the third arrangement appears to be quite inefficient; the second may at least have the virtue of cheapness.

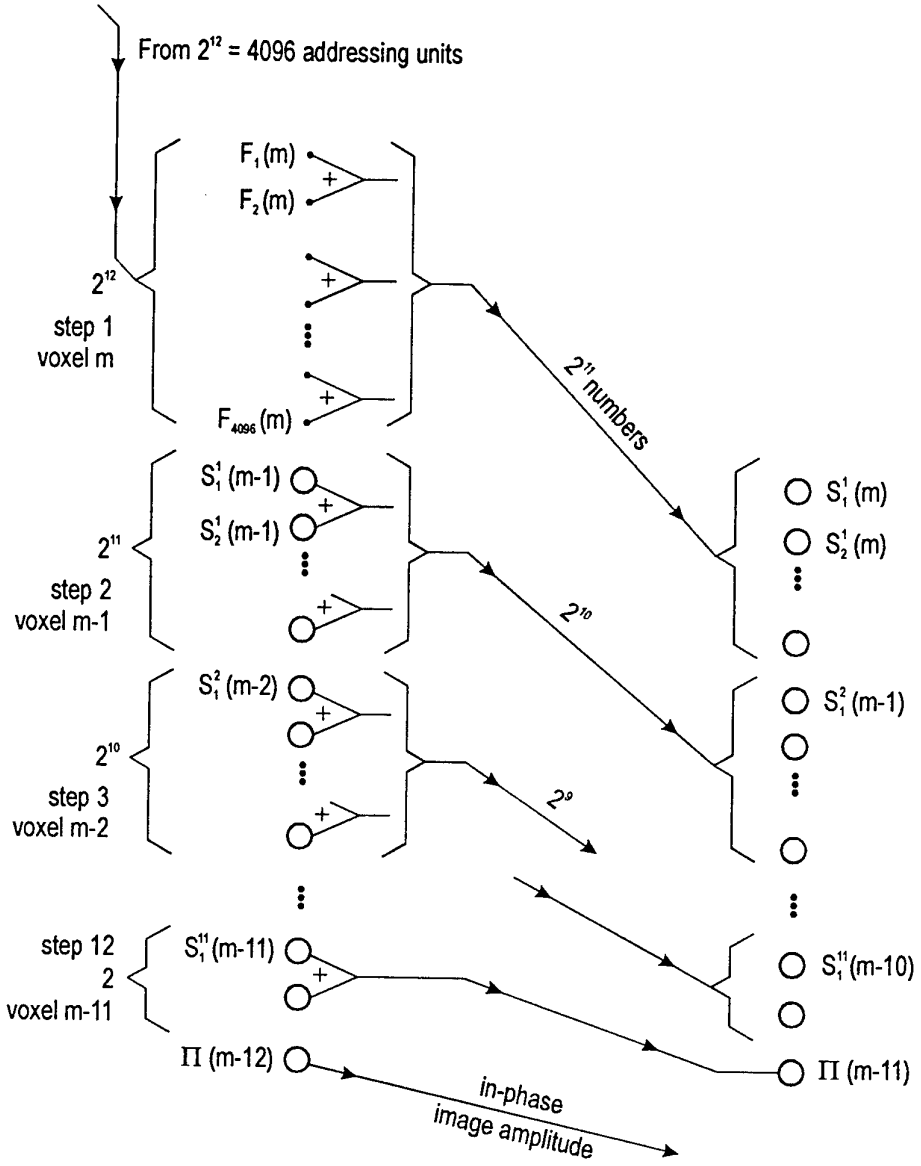

Figure 4: *The special summer. Each circle represents an address, the same address on the right as on the left. Progress from left to right represents one clock cycle; the clock maintains synchronisation between all operations. There are assumed to be* 2^{12} *or 4096 elements.* $F_n(m)$ *is the weighted voltage from the* n *th element to be used in forming the in-phase image amplitude* $\Pi(m)$ *for the* m *th voxel. The* S*s are partial sums formed during the calculation of* $\Pi(m)$*; thus* $S_1^q(m)$ *represents the sum of the first* 2^q *weighted voltages used in forming* $\Pi(m)$*. An identical summer, but with different inputs, handles the quadrature part.*

4. Calculation of Address

4.1 General

This Section 4 gives details of the addressing to be performed by the 2^{12} store-add-fetch units shown on the left of Figure 3.

As stated previously, from Equation (4), when the appropriate voltage is being retrieved, the calculation of the address is essentially the calculation of the round-trip time, given by Equation (5). It is assumed that no interpolation is performed, that is, when Equation (4) calls for a particular value of time, the nearest time at which a sample voltage was taken is used. Since samples are taken at the rate of 20 MHz and the central frequency f_c is 3.5 MHz, this lack of interpolation leads to a maximum phase error (at f_c) of ±30° approximately. This phase error is acceptable: the degradation of the image will be small but not negligible.

4.2 Expansion of the Round-Trip Distance

The expression (2) or (5) for the round-trip distance, namely

$$L(\mathbf{r}, n) = r + |\mathbf{r} - \mathbf{R}_n|$$
$$= r + \left(r^2 - 2\mathbf{r}\cdot\mathbf{R}_n + R_n^2\right)^{1/2}$$

(see Fig. 1), involves a square root. To compute the expression, via explicit calculation of the square root, is not all that bad, as the time taken for a square root on a typical serial computer is the same as for 6 to 7 multiplications. However, it is hoped to do better. With parallelism, one might reduce the time to that of one MAD operation.

As is well known, the expression for the return leg of the path can be expanded in a power series [see Steinberg 1976, Kino 1987, or particularly Ziomek 1985], the first two terms of which yield an expression valid in the far field. The procedure is to expand the square root on the right-hand side of (5) in powers of $\mathbf{R}_n$, using the binomial theorem:

$$(a+b)^{1/2} = a^{1/2} + \tfrac{1}{2}\frac{1}{a^{1/2}}b + \frac{\frac{1}{2}\left(-\frac{1}{2}\right)}{1\cdot 2}\frac{1}{a^{3/2}}b^2 + \frac{\frac{1}{2}\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)}{1\cdot 2\cdot 3}\frac{1}{a^{5/2}}b^3 + \cdots, \qquad (6)$$

valid when $|b| < a$. In the present case we put $a = r^2$, $b = -2\mathbf{r}\cdot\mathbf{R}_n + R_n^2$. (This choice of a and b is appropriate because, in an expansion starting from the far field, we must regard r, the range of the voxel, as large, and hence $\mathbf{R}_n$ as relatively small; see Fig. 1.) Note that the second term of b is already 2nd order in $\mathbf{R}_n$; this must be borne in mind when gathering together subterms of a given order to form the final expansion of L.

Let us define the ratios

$$\rho = \frac{R_n}{r}, \qquad \rho_r = \frac{\mathbf{R}_n\cdot\hat{r}}{r}. \qquad (7)$$

Both are related to the dimensionless vector $\mathbf{R}_n/r$: ρ is the magnitude of that vector, while ρ_r is the component of that vector along $\mathbf{r}$. After taking out r as a common factor, we find that the expansion is

$$L(\mathbf{r},n) = r[2-\rho_r+\tfrac{1}{2}(\rho^2-\rho_r^2)+\tfrac{1}{2}\rho_r(\rho^2-\rho_r^2)+\tfrac{1}{8}(\rho^2-\rho_r^2)(-\rho^2+5\rho_r^2) + \tfrac{1}{8}\rho_r(\rho^2-\rho_r^2)(-3\rho^2+7\rho_r^2)] \quad (8)$$
$$= r\left\{2-\rho_r+\tfrac{1}{2}(\rho^2-\rho_r^2)\left[1-\tfrac{1}{4}\rho^2+\tfrac{5}{4}\rho_r^2+\rho_r\left(1-\tfrac{3}{4}\rho^2+\tfrac{7}{4}\rho_r^2\right)\right]\right\}.$$

In Equation (8), consider the first of the two expressions for $L(\mathbf{r},n)$. Within the square bracket the expansion has been written, beginning with the zero order term, 2, through the first order term $-\rho_r$, and so on, up to and including the fifth order term. The combination $2-\rho_r$ in the square bracket gives the far-field expression. Note that ρ and ρ_r are both first order in $\mathbf{R}_n$, so that the expansion[3] in powers of $\mathbf{R}_n$ may be thought of as an expansion in powers of ρ and ρ_r. ρ and ρ_r are moderately small compared to unity.

We introduce x, y and z axes, with the z axis perpendicular to the plane of the array, as shown in Figure 5. Thus the position of the nth element is $\mathbf{R}_n = (X_n, Y_n, 0)$, while the image point is at $\mathbf{r} = (x, y, z)$. Then

$$\rho^2 = \frac{X_n^2+Y_n^2}{r^2}, \qquad \rho_r = \frac{X_n x + Y_n y}{r^2} = \frac{X_n u + Y_n v}{r}. \quad (9)$$

In the last expression for ρ_r, we have introduced the direction cosines of the image point, u and v, defined by

$$u = x/r, \qquad v = y/r. \quad (10)$$

The significance of these ratios is as follows. u is the sine of the angle between $\mathbf{r}$ and the yz plane (Fig. 5); similarly for v and the xz plane. To first order in angle, u and v are the angular coordinates of $\mathbf{r}$; they are angles directed away from the z axis in the direction of the x axis and y axis respectively.

[3] Note that the expansion is to be thought of as being done at fixed values of the angular coordinates of $\mathbf{R}_n$. Hence, to get the full pth order term, including all subterms, one has to go further into the expansion than one might expect. The subterm $-\rho_r^2$ in the above expression arises in this way. The pth order term is the sum of all subterms in the square bracket of the form $\rho^c\rho_r^d$, where $c+d=p$ (finally multiplied by the r out in front). The coefficients are all pure numbers.

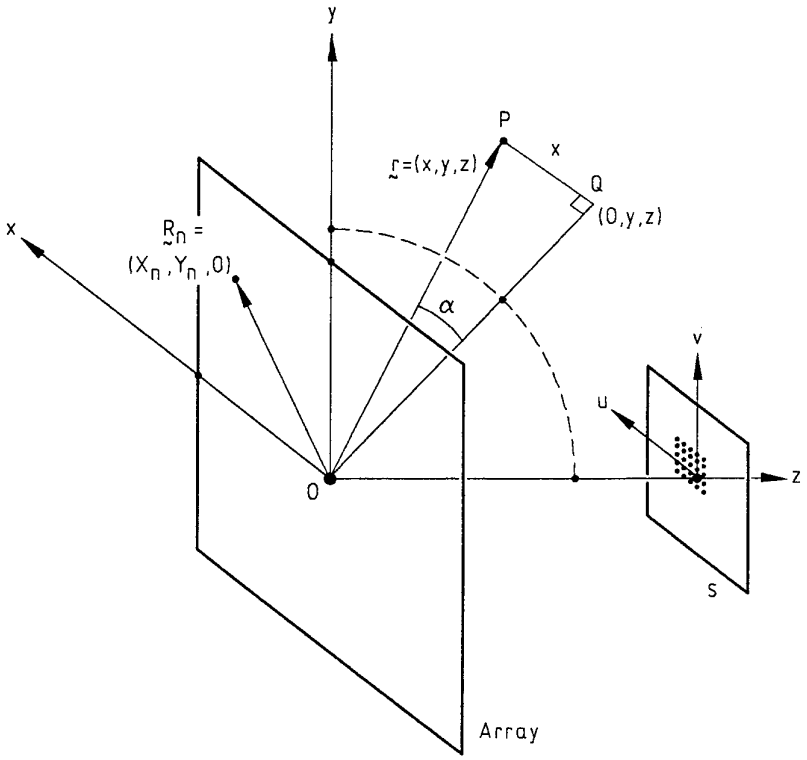

Figure 5: Coordinate system and direction cosines. The origin is at the centre of the planar array; the z axis is perpendicular to the array. P, the image point, when projected onto the yz plane, yields Q. The first direction cosine is $u = \sin\alpha$, where α is shown. The plane S is used in giving an exact geometrical interpretation of the grid of voxels, as explained in Section 4.3.2. For small angular deviations from the z direction, the voxels having a given value of r coincide very closely with the grid points shown.

4.3 Choice of Voxels

4.3.1 General

Because the distance L has a simpler dependence on the direction cosines u and v than on the spherical polar angles, it is advantageous to choose the coordinates of the voxels (their centres) to lie on a grid in which the values of u and v are stepped through in equal increments. (There seems to be no special virtue in using the angles—spherical polar angles—a choice which, in any case, treats x on a different footing to y.)

We therefore assume that each of r, u and v is stepped through in equal increments. (Note however that some computing time can be saved by stepping along the range direction by *factors* of 1.001, that is, taking the next value of r to be always 1.001 times the previous r value. Such multiplicative stepping is justified provided that the specification for the operational sonar is for a range resolution of 1 mm per m of range.) Thus we have

$$r = ka, \qquad u = lb, \qquad v = mc, \tag{11}$$

where a, b and c are the respective increments, and each of k, l and m runs through a set of consecutive integers.

4.3.2 Geometrical Interpretation

This section, not necessary to understanding the rest of the report, gives a geometrical interpretation of the grid of points (11), involving the plane S in Figure 5. From Equation (10) and the relationship $x^2 + y^2 + z^2 = r^2$, the Cartesian coordinates of a grid point (11) are

$$(x, y, z) = ka\left(lb, mc, \sqrt{1 - l^2b^2 - m^2c^2}\right),$$

where ka is a common factor. Consider the grid points on the sphere $r = ka = \text{constant}$ (a 2-D grid); and consider their orthogonal projection onto the plane $z = ka = \text{constant}$ (plane S) that is tangent to the sphere. Then the plane-projected grid is

$$(x, y, z) = ka\,(lb, mc, 1) = ka\,(u, v, 1),$$

where u and v are the direction cosines of the original grid points. This new grid is a rectangular grid, shown in Figure 5. The corresponding points of the original grid (11) are found by projecting from this grid, perpendicular to the plane S, onto the sphere.

Changing to a different value of k (different radius) is also easily visualised. A first way is to note that, when we change from k_1 to k_2, the grid points (11) on the k_2 sphere lie in the same direction from the origin as did the points on the k_1 sphere.

A second way is to note that, in such a change, we move from one plane S_1 to a new plane S_2, where the distance of the plane from the origin has been multiplied by k_2/k_1. At the same time, the size of the spacings in the plane-projected grid on S_2 is multiplied by this same factor. (In other words, the new plane-projected grid points are obtained by projecting the old ones via lines from the origin.) The new plane-projected grid is related to the corresponding grid on the new sphere in the same way as in the paragraph before last.

4.4 Polynomial Approximation

4.4.1 Choice of Fifth Degree

The truncation of the expression (8) for the round-trip distance $L(\mathbf{r}, n)$ at the end of a given order leads to a polynomial expression (see next paragraph). Calculations for extreme cases of the operational system show that, for a reasonable representation of the distance $L(\mathbf{r}, n)$, the terms up to and including the 5th order term are required. This is so because the 6th and 7th order terms give at most a contribution of 6° and 4° respectively to the phase change along the path at the central frequency, while the 5th order term gives a maximum contribution of 102°. (However, the conclusion regarding which terms are to be included is modified in Section 7, where individual subterms are considered.)

The expansion up to and including the 5th order term is a 5th degree polynomial in each of l and m separately. (That expansion is also a 5th degree polynomial in l and m jointly, in the sense that each subterm is a coefficient times $l^c m^d$ with $c \geq 0$, $d \geq 0$, $c+d \leq 5$).

4.4.2 Algorithm for Fast Calculation

To deal with the voxels, it is necessary to step through all the values of k, l and m in Equation (11). As discussed already (see Fig. 2), it is proposed that the outermost loop be over k (i.e. over r); and that the innermost loop be over m (i.e. over v). There are 1000 values of m.

Since the round-trip path L is, to a sufficient approximation, a 5th degree polynomial in m, *its fifth difference is a constant,* where 'fifth difference' will now be defined. For the moment, let L_m be any sequence of numbers, defined for $m = \cdots, -2, -1, 0, 1, 2, \cdots$. In the special case we have particularly in mind, L_m denotes the round-trip path L evaluated for the value m. The notation

$$\nabla L_m = L_m - L_{m-1}$$

denotes the *first backward difference* of L, evaluated at m [Burden and Faires 1985, p. 104]. The term 'backward' will now be dropped, since the only differences that will concern us will be backward ones. Also we write, for example,

$$\nabla L_{m+2} = L_{m+2} - L_{m+1}$$

for the first difference evaluated at $m+2$. Then the second difference (taken implicitly at m, unless otherwise stated) is

$$\begin{aligned} \nabla^2 L_m &= \nabla\left(\nabla L_m\right) = \nabla L_m - \nabla L_{m-1} \\ &= \left(L_m - L_{m-1}\right) - \left(L_{m-1} - L_{m-2}\right) = L_m - 2L_{m-1} + L_{m-2}\,. \end{aligned}$$

It is useful to imagine a column of values of L_m written out opposite the corresponding values of m. Then simple processing enables us to write in a further column the values of the first differences ∇L_m, each again written opposite the corresponding m. The procedure is readily extended to $\nabla^2 L_m$ and higher differences. If L_m is a polynomial in m, of degree p, say, simple algebra shows that the first difference ∇L_m is a polynomial of degree $p-1$. By repeating the procedure, we find that the second difference has degree $p-2$, and so on. The pth difference $\nabla^p L_m$ is a constant. For the *path length* L, it is then clear that the fifth difference $\nabla^5 L_m$ is independent of m (to a sufficient approximation). This leads to a rapid way of calculating L, enabling the calculation to be done in one clock cycle.

4.5 Hardware for Polynomial Approximation

We postpone to Section 4.6 the problem of initialising and finalising loops, particularly the innermost loop, over the voxels. For the present, therefore, we consider the progress of the calculation once the innermost loop is sufficiently under way so that 'the pipeline is filled'.

4.5.1 Calculation of Address; Fetching the Weighted Voltages

The fast method of calculating the 5th degree polynomial L is based on the properties of differences discussed in Section 4.4.2. Recall that the path length L_m is to be calculated for $m = 1,2,\cdots,1000$, where m determines the relevant voxel. The analogy of the table of values and differences (Section 4.4.2) is relevant here. Basically the method consists in first determining all the values in the table across one row—or, actually, across a diagonal sloping downwards to the right. Then it is a simple matter to fill in the entries that are one row down from those entries. The method involves repeating the process until L_{1000} has been calculated.

At the beginning of each step in the calculation, the most recent values determined are:

$$L_{m+1}, \nabla L_{m+2}, \nabla^2 L_{m+3}, \nabla^3 L_{m+4}, \nabla^4 L_{m+5} \text{ and } \nabla^5 L_{m+6},$$

for some m. We progress one step by performing the five additions

$$\begin{aligned} L_{m+2} &= L_{m+1} + \nabla L_{m+2} \\ \nabla L_{m+3} &= \nabla L_{m+2} + \nabla^2 L_{m+3} \\ \nabla^2 L_{m+4} &= \nabla^2 L_{m+3} + \nabla^3 L_{m+4} \\ \nabla^3 L_{m+5} &= \nabla^3 L_{m+4} + \nabla^4 L_{m+5} \\ \nabla^4 L_{m+6} &= \nabla^4 L_{m+5} + \nabla^5 L_{m+6}, \end{aligned}$$

and recognising that $\nabla^5 L_{m+7} = \nabla^5 L_{m+6}$ since the fifth difference is constant.

The method is represented in Figure 6, which shows the address and fetch operations of a single SAF (store-address-fetch) unit. In the figure, progress from left to right represents one time-step (clock cycle) within the loop over m (Fig. 2); each register on the right represents the same register as on the left, but in general the contents have changed. The figure is drawn for a time-step not near the beginning or the end of the loop over m, so that initialisation can be ignored. In the time-step, each difference of L, from the fifth to the first, is used to update the value of the next-lower difference. In particular L itself (zeroth difference) is updated; the 5th difference needs no updating.

The registers containing L and its differences are shown explicitly in the diagram of the overall arrangement, Figure 2 (on the left-hand side).

It is assumed that, for faster computation, the machine's representation of L is scaled so that the operations are performed in integer arithmetic. In addition, a constant offset is built into L so that the machine's L may be converted to the relevant address (see below) by a very simple operation, such as removing the last few digits. The simple operation will be called the 'truncate' operation.

L, after the truncate operation, becomes the address from which the in-phase weighted voltage is fetched (Fig. 6). The quadrature voltage requires a constant offset to be added onto the in-phase address. (To save adding one to the length of the pipeline, the quadrature address would be calculated directly from L—not from the truncated in-phase address. It is assumed that the direct calculation can be done in one clock cycle.)

4.5.2 The Pipeline

The length of the pipeline in the addressing unit is readily determined from Figure 6, with the convention adopted in Section 3.5 that the final pair of fetches are counted as parts of the two respective summers. The pipeline length is 6 cycles, as is seen from the fact that there is a one-to-one correspondence between an operation (or pair of operations) and a label on the right, where those labels run from $m+1$ to $m+6$.

The number of operations being performed simultaneously is different from the pipeline length and may be estimated as follows. There are: (i) five adds for the calculation of the polynomial; (ii) one truncate; and (iii) one operation conceptually consisting of two parts: an addition to obtain the quadrature address, and a truncation of that address. (The two final fetches are again counted as part of the summers.) We conclude that there are 7 simultaneous operations.

The estimation, for a *serial* computer, of the number of operations required *per element-voxel pair* proceeds similarly, with the same result, 7. But in this case, because the summation over elements is done in the conventional way, it is possible to express the result slightly differently as follows. With a total of 9 operations per element-voxel pair, the system can perform the two summations as well as the addressing.

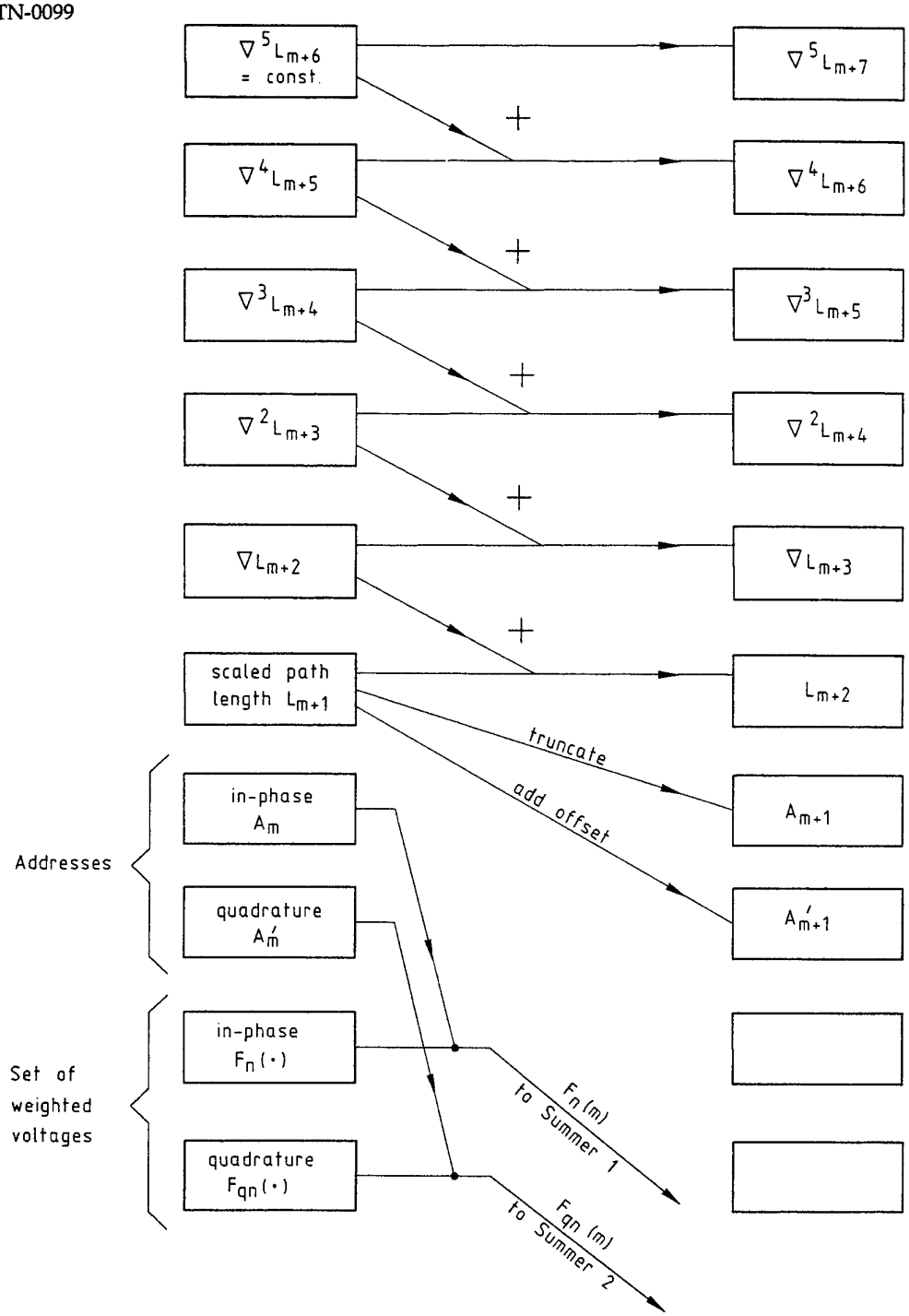

Figure 6: *Addressing procedure of the store-address-fetch (SAF) unit for the n th element; altogether there are approximately 4000 such units. Each box represents an address (or a set of addresses), the same address on the right as on the left. Progress from left to right represents one clock cycle; the clock maintains synchronisation.* L_m *is the scaled round-trip path length for the m th voxel.* ∇ *is the backward difference operator (with respect to m). The outputs are a pair of weighted voltages.*

4.6 Initialising and Finalising the Loop

Special attention must be paid to the initialisation of the loop over m (i.e. over v) because it affects the simplicity of the SAF units, and therefore the financial viability of the whole proposed computing scheme.

4.6.1 Finalising

Let us approach the problem in steps, beginning with finalisation of the loop (Figs. 2 and 3). The finalisation is easily achieved by having the master computer count the appropriate number of clock cycles since the parallel computer started the pipeline part, or looping part, of its procedure. It is assumed that the two computers become synchronised with each other at the start of the loop (end of initialisation of the loop over v, see below). Starting from that time, it is known how many cycles there are (1014 in fact, see Section 5.1) until the 1000th desired intensity is transferred to the master computer. The fact that computations may continue thereafter for a while in the parallel computer, attempting to calculate voxel intensities outside the volume to be imaged, does not matter; the master computer ignores such values.

4.6.2 Initialising; Initialising the Path Length

For the pipeline procedure to begin, L and its five differences, for the initial value of m, must be set up. Once this setting-up is done, and the operations (mainly additions) are set going in the whole pipeline, everything proceeds as desired in the parallel computer. It is necessary only for the master computer to ignore the first few outputs (in fact, to ignore the first 14 outputs after the end of initialisation, see Section 5.1), accepting the next 1000.

Three aspects of initialisation need to be considered. The first is the initialisation of L and its five differences. One way is to have each SAF unit act as a serial computer for this phase. The number of multiplications or additions (MADs) required can be reduced slightly, first, by noting that is sensible to make $b = c$ in Equation (11), and second, by transferring some of this work to the master computer, namely an add, a divide and two multiplies. (These few computations could be done while the previous pipeline calculation was proceeding.) Then the SAF unit needs to do 5 MADs to calculate ρ and ρ_r (Eqns 9 and 10), then 30 MADs to calculate a 5th degree, a 4th degree, etc. polynomial for L (Eqn 8) and its first, second, etc. differences, and finally 6 MADs for the final multiplication by r (times a constant) for each polynomial (see Eqn 8), yielding a total of 41 operations. At 41 clock cycles, this time is very small compared to the 1000 or more clock cycles needed for the loop over m. (It may be possible to reduce the number of operations below 41.)

4.6.3 Synchronising

The second aspect of initialisation concerns synchronising of the master with the parallel computer at the start of the loop. This may be done as follows. When the master computer feeds the last parameter to the parallel computer, it begins counting cycles. It is known how many cycles the parallel computer needs to complete initialisation, and it is known that in the 15th subsequent cycle the first desired intensity flows from the end of the pipeline. At this time the master computer accepts its first voxel intensity value; and 999 cycles later, it accepts its last value and then takes control, ending the loop. (An alternative arrangement is that, once the initial value for L and its differences have been set up, the parallel computer signals that it is ready and immediately starts the pipeline procedure. Upon receipt of the signal the master computer begins counting cycles.)

4.6.4 Switching Operations On

The third aspect of initialisation concerns internal controls in the parallel computer: switching the adds and other internal operations of the pipeline on and off. As pointed out above, the pipeline operations that take the value of L (not the first and higher differences) and process it further, including the fetching of the weighted voltages and the additions in the summers, can proceed indefinitely with no harm. However, at initialisation, it is necessary to stop the additions that occur into five of the six L registers (Fig. 6). One way is to stop those five additions for five cycles while L and its first to fourth differences are fed serially into those registers. (The fifth difference can be fed in one cycle earlier.)

An alternative is to feed in the 4th difference, the 3rd difference, etc. in succession, while disabling for one cycle the corresponding addition that feeds into the same L register. A third option is to feed L and its five differences simultaneously into their registers, the additions being disabled for that one cycle. (Note that, with any of the three methods, the additions themselves do not have to be stopped for the one or five cycles, as long as the value fed in directly is made to override the result of the addition.)

5. Performance

5.1 Total Pipeline

Except for the initialisation phase, the whole procedure for the benchmark method is pipelined so that when the time advances by one clock cycle, a new total image intensity value for one voxel issues from the end (Fig. 3). Several voxels are being processed at once: the pipelining extends from the calculation of the fourth difference of the $(m+6)$th voxel as the step at the beginning of the pipeline (Fig. 6), through the summation (Fig. 4) to the calculation of the total intensity of the $(m-13)$ th voxel as the

step at the end of the pipeline (Fig. 3). Thus the total pipeline is 20 operations in length (13+6+1, since the operations at the two ends must both be included). The breakdown of this length by components is of interest. The addressing contributes length 6, the summers contribute 12 and the square-and-add 2, for a total of 20. (In the square-and-add unit, we include for counting purposes an initial fetch from the summers and a final store from the display unit—see the method of Section 3.5.)

On the other hand, once the loop has been initialised, *fewer* than 20 cycles are needed until the first desired intensity flows out. This is so because the initial L has already been obtained (Fig. 6). Consequently the pipeline effectively begins with the truncation and the adding of the offset to obtain the $(m+1)$th addresses. Thus 5 steps in the pipeline are 'eliminated'; the first desired intensity flows out 15 cycles after the last cycle of initialisation.

5.2 Performance: Computing Time

Let us concentrate on the time for beamforming proper, and recall that one voxel intensity is produced, on average, in one clock cycle. Consequently, for a 1 Gflops parallel computer—a machine with 10^9 clock cycles per second —the image of 3×10^9 voxels is produced in 3 seconds. Allowing a 'factor of safety' of 2 to allow for overheads in a practical computer (Section 2.4), we reach an estimate of 6 s for the total computation time.

The other process that might take appreciable time is the dechirp/transform (Section 2.3). This stage is discussed in Section 6.4.

5.3 Hardware Requirements

This section attempts to describe the hardware of the parallel computer in a manner that is relevant to cost. It is assumed that, to allow the greatest computing speed, the calculations are performed in integer arithmetic on scaled numbers, though the implementation of this feature has not been worked out.

Each of the two summers contains 4095 storage locations and performs 4095 additions simultaneously. There is no programming complexity, for exactly the same operations are performed from one cycle to the next. This remains true even during initialisation of each loop over v (since what the summers produce then is ignored). It appears that the cost is not inherently high, but could be high in practice due to the lack of a mass market.

Regarding the 4000 SAF units, note first that the storage capacity needed in each one is not great: about 340×10^3 real numbers (340 from 80+131+131, from Section 6.3, assuming the raw signals are to be retained). Consider first the steady-state pipeline phase (Fig. 6) (the initialisation phase being ignored for the moment). Then the instructions are entirely repetitive. Several operations (about 9) are occurring simultaneously; but as there is virtually no controlling logic involved, this phase, despite its parallelism, should not be inherently expensive.

Second, during the initialisation phase, the SAF unit is to run as a serial computer, executing 41 MAD operations. During that phase, the only logical operations are the deactivation, followed by the activation, of the five additions involving L and its differences (Fig. 6). The program for the initialisation is therefore simple. Note that the parallel computer runs the same program (initialisation plus pipeline) on each occasion on which control is passed to it. Again the SAF units seem not inherently expensive to make, even given the large number (4000) of them; the lack of a mass market could however prove a difficulty.

A significant feature of the fifth difference method is that roundoff errors accumulate in an exponential fashion. In particular, an initial error of ε in the 5th difference of L, is readily shown, after 1000 steps of v, to lead to an error in L itself approximately equal to

$$(1000)^5 \varepsilon/5! = 8.3 \times 10^{12} \varepsilon .$$

Despite this huge growth in error, the method is still viable. A closer examination (including the number of bits required for other purposes) shows that the only effect is on the number of bits required. Whereas 32-bit integer arithmetic suffices in most signal processing work, the present application would require close to, but not more than, 64-bit integer arithmetic.

6. The Dechirp/Transform Stage

6.1 Effect of Absorption

The discussion in the body of Section 6 will ignore the effect of absorption in the medium, which introduces a factor into $E_n(t)$ that depends on both frequency and path length. This effect threatens to disrupt the scheme in which dechirping is done by a simple crosscorrelation. Dechirping might then take longer than the times estimated below based on the use of the FFT. Worse, it is conceivable that dechirping might not be possible at all.

Towards solving these problems, an investigation (not presented here) appears to show that the idea of breaking up the voltage stream into segments of suitable length[4], together with applying in each segment an approximation to the exponential absorption factor, reduces the computation to essentially the same length as in the absence of absorption. The absorptive effect will henceforth be ignored.

[4] This idea of segments is the same as that described at the end of Section 5.4.3.

6.2 Mathematics of the Dechirp/Transform Stage

By the dechirp/transform stage (Section 2.3) we mean the stage in which dechirping, the production of the quadrature part, and weighting are carried out. The overall results, for the in-phase and quadrature part respectively, are

$$E_{rn} = w_n \xi \otimes E_n \tag{12}$$

$$\begin{aligned} E_{rqn} &= -w_n \hat{\xi} \otimes E_n = w_n \mathcal{H}(\xi \otimes E_n) \\ &= w_n \xi \otimes \hat{E}_n \end{aligned} \tag{13}$$

(apart from a constant multiplying factor which may be dropped in this context). Here ξ, E_n, E_{rn} and E_{rqn} are functions of time t: $E_n(t)$ is the in-phase voltage stream, $\xi(t)$ is the effective voltage input into the projector (effective, due to the fact that the response of the transducers is not constant over frequency) and w_n (independent of t) is the weighting of the nth element. The encircled cross denotes complex crosscorrelation, defined by

$$(f \otimes g)(t) = \int_{-\infty}^{\infty} f^*(t') g(t + t')\, dt'$$

(f^* being the complex conjugate of f). For future use the general or complex definition of crosscorrelation is given here, even though all the quantities in Equations (12) and (13) are real. The notations $\hat{f}$ and $\mathcal{H}(f)$ both denote the Hilbert transform of the function $f(t)$, defined by

$$\hat{f}(t) = \frac{1}{\pi} \mathrm{P} \int_{-\infty}^{\infty} \frac{1}{t - t'} f(t')\, dt',$$

where P denotes the principal value of the integral. As discussed in Section 2.3, $\hat{f}(t)$ is essentially $f(t)$ but with the phase of each Fourier component shifted by 90°. The above concepts are discussed by Rihaczek [1985, pp. 15–20]. The equality of the three expressions for E_{rqn} follows from the Hilbert transform properties of convolutions and will be shown in a forthcoming report by the author.

Some further results will prove useful for optimal computation. Adding Equation (12) to $-j$ times (13), we obtain

$$E^*_{rcn} = w_n \xi_c \otimes E_n, \tag{14}$$

where $\xi_c = \xi + j\xi_q$ denotes the complex or analytic signal, and similarly $E_{rcn} = E_{rn} + jE_{rqn}$. Let us denote the Fourier transform of a function $g(t)$ by

$$g(f) = \int_{-\infty}^{\infty} g(t)\, e^{-j2\pi f t}\, dt.$$

Then, in the frequency domain, Equation (14) becomes

$$E^*_{rcn}(-f) = w_n \xi^*_c E_n, \tag{15}$$

where each of ξ_c, E_n and E_{rcn} is now a function of frequency.

6.3 Number of Operations

We estimate the number of operations required in the dechirp/transform stage, for a voltage stream $E_n(t)$ of length N. Two basic methods will be explored: first, the use of the straightforward, i.e. non-Fourier, method of evaluating of the crosscorrelation integral or sum (and the Hilbert transform), and second, the use of the fast Fourier transform (FFT).

6.3.1 'Straightforward' Method of Computation

In the straightforward method to be described, the author has attempted to optimise the method; it is difficult to see how further optimisation could make more than a small difference. First, for each element, the time-stream of values $w_n\xi(t)$, rather than the stream $\xi(t)$ itself, is stored, for efficient computational use of Equation (12). Second, $-w_n\hat{\xi}(t)$ is similarly stored; the quadratic part $E_{rqn}(t)$ is then computed by a second crosscorrelation rather than by a numerical Hilbert transform (i.e. the first of the three expressions for E_{rqn} in Eqn 13, rather than the second, is used). The method requires the following steps:

1. Crosscorrelation of $w_n\xi$ with E_n. Number of operations: $2PN$. Here P is the length of, or number of samples in, the projected signal $\xi(t)$. Note that P is much less than N (the length of the received voltage stream), since P corresponds typically to 1000 cycles ($P = 5714$ samples), while N corresponds to 14 000 cycles ($N = 80\,000$ samples, see Section 2.1). In the above expression for the number of operations, we have counted only each multiplication and its addition into the sum, ignoring the incrementing of the variable of integration.
2. Crosscorrelation of $-w_n\hat{\xi}$ with E_n. Operations: $2PN$.

The total number of operations is thus $4PN$.

6.3.2 FFT Method of Computation: Main Description

We turn to the FFT method. Note that the FFT, performed on a vector of length $N = 2^n$, with n an integer, requires $(N/2)\log_2 N$ complex multiplications and $N\log_2 N$ complex additions-or-subtractions [Bergland 1969]. Since the number of real MAD operations is 6 for a complex multiplication and 2 for a complex addition, the total number of real operations is $5N\log_2 N$. (This is perhaps a pessimistic estimate when the original vector is real, as it is in the first step below.)

Again the author has attempted to optimise the method, and it is difficult to see how further optimisation could make more than a small difference. This is especially true in respect of the dominant or logarithmic term (see below), since both a forward and an inverse FFT are required. The resulting method is as follows:

1. FFT the in-phase voltage stream $E_n(t)$ to produce $E_n(f)$. Number of real operations: $5N\log_2 N$.
2. Now apply Equation (15). For each element, the product $w_n\xi_c^*$ in the frequency domain (actually a vector over f) is permanently stored. For each f, multiply this quantity by $E_n(f)$ obtained from step 1. This gives the crosscorrelated (i.e. dechirped) analytic signal $E_{rcn}(f)$ in the form $E_{rcn}^*(-f)$. Operations: $3N$. (The number of operations is not $6N$, because half the frequency components of an analytic signal are zero.)
3. Perform an inverse FFT to yield, from (14), $E_{rcn}^*(t) = E_{rn}(t) - jE_{rqn}(t)$. Hence the real and imaginary parts yield $E_{rn}(t)$ and $E_{rqn}(t)$, which are the in-phase and quadrature dechirped signals. Operations: $5N\log_2 N$.

Thus the total number of real operations is $10N\log_2 N + 3N$.

6.3.3 FFT Method of Computation: Further Discussion

Savings beyond the figure of $10N\log_2 N + 3N$ operations just calculated may be possible. First, by the fact that the vector to be transformed in step 1 is real. And second, because the chirp is much shorter than the received voltage stream, the stream may be broken up into segments, each longer than a chirp. Each segment would be dechirped by the above procedure. The segments would have to overlap, because otherwise the required integral would on occasion extend outside the segment.

To investigate this possibility, let P be the number of samples in a chirp (as before), and let the segment be of size θP, where $\theta > 1$. The total number of operations required is now

$$N_{op} = \frac{N}{(\theta-1)P}\left[10(\theta P)\log_2(\theta P)\right].$$

To obtain this expression, first, in the result for the 'total number of real operations' in Section 6.3.2, we drop the small term '$+3N$' and replace N by θP to get the operations required in one segment; this yields the square bracket in the expression. This square bracket must then be multiplied by the number of segments; allowing for overlap, this number is $N/[(\theta-1)P]$; hence the expression for N_{op}.

N_{op} can then be minimised by putting $\partial N_{op}/\partial\theta = 0$; details that make the calculation more efficient are omitted here. For $N = 80\ 000$ and the typical value $P = 6000$, it is found that the number of operations is minimised at about $\theta = 14$. Since θP is then roughly equal to N, N_{op} reduces to its nonsegmented value and so the savings by segmentation are essentially nil in this case. On the other hand, for the

smaller value $P = 1500$, the savings in operations amount to 12% (compared to $P = 6000$ as the standard). Note that the 'small' value $P = 1500$ still corresponds to a chirp containing very many cycles (Section 2.1). Thus, there is a range of such Ps, lying below the value 6000, for which segmentation yields small but appreciable savings.

The length of the voltage vector is 80 000, but the appropriate value of N for the FFT is greater than that, for two reasons. First, in a cyclic arrangement such as in the FFT, a convolution or crosscorrelation of two vectors is performed correctly only if the length N of the vectors after padding with zeros is greater than or equal to the sum of the two unpadded lengths. Assuming a chirp duration of 500 cycles of the central frequency, the unpadded length of the chirp comes to about 2900, pushing the required value of N up only slightly to 82 900. Second and more importantly, we must round the resulting value of N up to the next power of 2, namely $N=2^{17}=131\ 072$, assuming that the intermediate values of N yield FFT algorithms that are less fast.

6.3.4 Comparison of the Two Methods

The number of operations per sample point has been found to be $4P$ for the straightforward method and $10\log_2 N + 3$ for the FFT method. For the 'typical' value $N = 80\,000$ (Section 2.1), the FFT figure becomes 173. Therefore the straightforward method achieves equality when $4P = 173$, i.e. $P = 43$, and is worse for larger P. For the typical figure of $P = 6000$, the straightforward method requires 140 times as many operations than the FFT method. For our problem the FFT method wins.

For the FFT method, on a 1 Gflops serial machine, the time to treat 4000 elements comes to 94 s or, allowing a 'factor of safety' of 2, to 3.1 min. This time is unsatisfactorily long—a problem that will be addressed in Section 6.4.

6.4 Selection of Hardware for the Dechirp/Transform Stage

We saw in Sections 2.3 and 6.3 that, if both stages (dechirp/transform and beamforming proper) are performed by equally fast serial machines, the dechirp/transform takes only a tiny fraction of the total time. It can therefore be anticipated with a fair degree of confidence that, if parallel architecture is used, and *equal costs are expended* on the two stages, the dechirp/transform stage will take a much smaller computing time than the beamforming. Second, it may be anticipated that if the design is optimised subject to the constraint that the two stages take approximately *the same computing time,* the dechirp/transform will be by far the cheaper. And third, we may expect that an overall optimum can be achieved in which the dechirp/transform is both considerably cheaper *and* considerably faster than the beamforming proper.

The finding of any of these optima depends on a detailed analysis, in which account is taken of available components and their cost. Here we content ourselves with describing two methods by which the computing time can be made considerably less than that of the beamforming proper.

One method is to increase the capability of each SAF unit so that it can perform a crosscorrelation via the FFT; then each element has its signal dechirped by its own SAF unit. This method achieves the computing time goal very easily because, even if each SAF unit acts here as a serial machine, the time taken, from Section 6.3, is (3.1 min)/4000 or 0.047 s. For purposes of exposition, this method will be taken as the primary one of the two; it is the method illustrated in Figure 2 and leads to a total computing time equal to that for the beamforming proper (6 seconds, Section 5.2).

But it may be that the first method increases the capital cost of each SAF unit by a factor of (say) 2, to achieve what is only a tiny part of the total computation. Actually the computing time of 0.047 s may be permitted to rise up to (say) one-half of the time of the beamforming proper, and still be more or less in balance. This fact prompts us to describe a second method, which appears to be cheaper. This method is to have a number of serial computers (small compared to 4000) acting in parallel with each other, each performing the dechirp/transform for a number of elements. Let us calculate the number of computers required on the basis that the dechirp/transform is to be done in half the time of the beamforming proper, that is , half of 6 s (Section 5.2). Then the number of 1 Gflops computers required to be acting in parallel is $3.0/0.047 = 64$; and each computer processes $4000/64 = 63$ elements.

7. Future Work

We draw attention to two aspects of addressing suitable for future work. Both could lead to small improvements in cost.

First, the far-field expansion of the path length $L(\mathbf{r}, n)$ to 5th order (Eqn 8), denoted by L_5, may be written as

$$L_5 = M_0 + M_1 + M_2 + M_3 + M_4 + M_5,$$

where, for example,

$$M_5 = \tfrac{1}{8} r\rho_r\left(\rho^2 - \rho_r^2\right)\left(-3\rho^2 + 7\rho_r^2\right)$$

is the 5th order term. Let us consider what happens when the expression for ρ^2 in (9), and the last of the two expressions for ρ_r in (9), are substituted into Equation (8) for L_5. Note that *the contribution to* L_5 *that is proportional to* v^5 (the term that is 5th degree in v), denoted by N_5, *is not the same as* M_5. Indeed N_5 is a subterm of M_5, namely

$$N_5 = \tfrac{1}{8} r(-7)\left(\frac{Y_n v}{r}\right)^5.$$

By finding an upper bound on $|N_5|$, it may be possible to show that N_5 can be dropped from the expression L_5 used for the path length, to sufficient accuracy. In that case, *a fourth difference method* can be used in place of the fifth difference method described in this report. This fourth difference investigation was not carried to completion, because the principal features of the scheme in this report would be preserved in the fourth order scheme. The main changes would be that the pipeline

length would be reduced by one, and the number of bits required in the arithmetic would be reduced from 64, though not to a value as low as 32.

We turn to a second item of possible future work. We cannot rule out the possibility that there are other methods for computing the address that are similar to the fifth (or fourth) difference method described in this report. In particular, let us define the *'span'* as the number of steps in v for which the path length L is computed by successive additions based on an initial set of values of L and its differences. The method we have described uses a span of 1000, corresponding to the entire range of values of v to be dealt with. There is no reason why *a span of less than* 1000 cannot be used, with L and its differences being recalculated from scratch at the end of each traverse through one span.

8. Conclusion

We have taken the problem of producing a 3-D image of 3×10^9 voxels from an acoustic array of 4000 sensor elements in a time of the order of seconds. It has been shown that, using computing elements with a clock speed of no more than 1 GHz, with the aid of parallelism and pipelining, the task can be accomplished in less than 10 seconds.

The key requirement of the computing scheme is the implementation of 44 000 adders (or devices of similar complexity) which, acting simultaneously, act as two special summers (4000×2 of the adders) together with 4000 store-address-fetch units (the remaining 4000×9 adders or their equivalent). A pipeline is set up that is 20 operations in length. The memory requirements, based on integers occupying 8 bits—both for the voltage inputs and the voxel outputs—consist of 640 megabytes for the input and computation and 3 gigabytes for the voxels.

The computing scheme developed is *scalable*, in that a halving of the clock rate can be compensated for by a doubling of the number of computing elements; while on the other hand, a doubling of the clock rate can compensate for a halving of the number of computing elements. In the former case, one could duplicate the stored voltages and have the computer calculate two voxels at once; one voxel would draw on one set of stored voltages and one-half of the computer elements. In the latter case, one could progress through the innermost loop over voxels, but summing over just half the sensor elements, and then redo the loop, summing over the remaining elements. The scheme is of course also scalable in the straightforward sense: reducing the clock speed to one-third triples the time taken.

From among the many options presented by scalability, the one discussed in this report has been chosen for two reasons. First, regarding 'straightforward' scalability, the chosen speed of 1 Gflops is a simple figure that is near the limit of current technology. Second, regarding the 'compensatory' scalability, the schemes obtained by scaling either to faster or slower clock speeds both involve a complication in regard to the dividing-up of the work.

9. Acknowledgements

Ian S.F. Jones made a number of valuable suggestions and is thanked for his inspiring leadership. The team drawn from the staff of GEC-Marconi Systems and CSIRO Ultrasonics Laboratory, exemplified by Ian G. Jones and David Robinson, contributed greatly to the success of the innovation program to date. John Shaw, Mike Bell and Donald McLean contributed some of the ideas in this report. The task manager was Jim Thompson.

10. References

Bergland, G.D. (1969). A Guided Tour of the Fast Fourier Transform. *IEEE Spectrum*, **6**, pp. 41-52.

Blair, D.G. and Jones, I.S.F. (1997). *Underwater Acoustic Imaging: Rapid Signal Processing.* (DSTO Technical Note DSTO-TN-0098). Melbourne: Aeronautical and Maritime Research Laboratory.

Burden, R.L. and Faires, J.D. (1985). *Numerical Analysis* (3rd Ed.). Boston, Mass.: Prindle, Weber and Schmidt.

Jones, I.S.F. (1996). *Underwater Acoustic Imaging Innovation Program.* DSTO Technical Note DSTO-TN-0065. Melbourne: Aeronautical and Maritime Research Laboratory.

Kino, G.S. (1987). *Acoustic Waves: Devices, Imaging, and Analog Signal Processing.* Englewood Cliffs, N.J.: Prentice-Hall.

Knudsen, D.C. (1989). A New Beamformer for Acoustic Imaging. In: *Oceans '89: An International Conference Addressing Methods for Understanding the Global Ocean.* New York: IEEE Press.

Rihaczek, A.W. (1985). *Principles of High-Resolution Radar.* Los Altos, Calif.: Peninsula Pub.

Steinberg, B.D. (1976). *Principles of Aperture and Array System Design—Including Random and Adaptive Arrays.* New York: John Wiley.

Ziomek, L.J. (1985). *Underwater Acoustics: A Linear Systems Theory Approach.* New York: Academic Press.

DISTRIBUTION LIST

Underwater Acoustic Imaging:
A Computing Hardware Approach to Rapid Processing

David G. Blair

(DSTO-TN-0099)

AUSTRALIA

DEFENCE ORGANISATION

Task Sponsor

S&T Program

Chief Defence Scientist
FAS Science Policy } shared copy
AS Science Corporate Management
Director General Science Policy Development
Counsellor Defence Science, London (Doc Data Sheet)
Counsellor Defence Science, Washington (Doc Data Sheet)
Scientific Adviser to MRDC Thailand (Doc Data Sheet)
Director General Scientific Advisers and Trials/Scientific Adviser Policy and Command (shared copy)
Navy Scientific Adviser (3 Copies)
Scientific Adviser - Army (Doc Data Sheet and distribution list only)
Air Force Scientific Adviser
Director Trials
Alan Burgess, ESRL Salisbury

Aeronautical and Maritime Research Laboratory
Director

Chief, Maritime Operations Division
Dr A Theobald (Research Leader)
Dr B Ferguson, MOD Sydney
Dr ISF Jones (Task Manager) (5 copies)
Dr D Wyllie
Mr JL Thompson, 52 Stokes Ave, Asquith, NSW 2077
Stuart Anstee, MOD Sydney
Ranjit Thuraisingham, MOD Sydney
John Shaw, MOD Sydney
Mike Bell, MOD Sydney
Doug Cato, MOD Sydney
John Riley, MOD Salisbury
Garry Newsam, MOD Salisbury
David Leibing, MOD Salisbury
Ross Barrett, MOD Salisbury
Warren Marwood, MOD Salisbury
David Blair, MOD Sydney 20 copies

DSTO Library
Library Fishermens Bend
Library Maribyrnong
Library Salisbury (2 copies)
Australian Archives
Library, MOD, Pyrmont (2 copies)
Library, MOD, HMAS Stirling

Capability Development Division
Director General Maritime Development (Doc Data Sheet only)
Director General Land Development (Doc Data Sheet only)
Director General C3I Development (Doc Data Sheet only)

Navy
SO (Science), Director of Naval Warfare, Maritime Headquarters Annex, Garden Island, NSW 2000 (Doc Data Sheet only)
Mine Warfare Systems Centre, PD
MHI, PD
MHC, PD

Army
ABCA Office, G-1-34, Russell Offices, Canberra (4 copies)
SO (Science), DJFHQ(L), MILPO Enoggera, Queensland 4051 (Doc Data Sheet only)
NAPOC QWG Engineer NBCD c/- DENGRS-A, HQ Engineer Centre Liverpool Military Area, NSW 2174 (Doc Data Sheet only)

Intelligence Program
DGSTA Defence Intelligence Organisation

Corporate Support Program (libraries)
OIC TRS, Defence Regional Library, Canberra
Officer in Charge, Document Exchange Centre (DEC), 1 copy
*US Defence Technical Information Center, 2 copies
*UK Defence Research Information Centre, 2 copies
*Canada Defence Scientific Information Service, 1 copy
*NZ Defence Information Centre, 1 copy
National Library of Australia, 1 copy

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy
Library
Head of Aerospace and Mechanical Engineering
Deakin University, Serials Section (M list), Deakin University Library, Geelong, 3217
Senior Librarian, Hargrave Library, Monash University
Librarian, Flinders University

OTHER ORGANISATIONS

NASA (Canberra)
AGPS
Mr Francois Duthoit, Head of Engineering, Thomson Marconi Sonar,
274 Victoria Rd, Rydalmere, NSW 2116 (4 copies)
Dr D McLean, Ultrasonics Laboratory, CSIRO, National Measurement Laboratory,
Bradfield Rd, West Lindfield, NSW 2070 (3 copies)

Dr Ling Guan, Dept of Electrical Engineering, University of Sydney, NSW 2006
Mr Alex Au, Ocean Technology Group, JO5, University of Sydney, NSW 2006

OUTSIDE AUSTRALIA

ABSTRACTING AND INFORMATION ORGANISATIONS

INSPEC: Acquisitions Section Institution of Electrical Engineers
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
Materials Information, Cambridge Scientific Abstracts, US
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS

Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US

SPARES (10 copies)

Total number of copies: 106

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Underwater Acoustic Imaging:A Computing Hardware Approach to Rapid Processing

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

David G. Blair

5. CORPORATE AUTHOR

Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Vic 3001 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TN-0099	AR-010-281	Technical Note	September 1997

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
5210/207/0714	ADS 94/195	DGMD	38	9

13. DOWNGRADING/DELIMITING INSTRUCTIONS

None

14. RELEASE AUTHORITY

Chief, Maritime Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Signal processing; Beam forming; Sonar arrays; Parallel processing; Real time operations

19. ABSTRACT

High-resolution underwater acoustic imaging using multi-element arrays implies a large computational load. For a three-dimensional viewing volume resolved into 3×10^9 voxels (volume pixels), with 4000 elements, the computations needed are around $9\times(1.2\times10^{13})$ floating-point operations. This report develops one of the more promising options for computing the full image. First, parallel computation is used to deal with the different sensor elements simultaneously, when calculating the address of the appropriate instantaneous voltage at the sensor element—or, equivalently, the calculation of the round-trip distance travelled by the acoustic pulse. This calculation requires, in a typical near-field situation, the computation either of a square root or of a fifth degree polynomial. This polynomial allows increased parallelism. Second, the summation in the beamforming is likewise done with a high degree of parallelism. A machine with the above design, with 10^9 clock cycles per second, would compute the entire image in roughly 6 seconds. Cost and availability are not investigated.